Web Mining and Recommender Systems

Text Mining
Learning Goals

- Introduce the topic of text mining
- Describe some of the difficulties of dealing with textual data
What kind of quantities can we model, and what kind of prediction tasks can we solve using text?
Prediction tasks involving text

Does this article have a positive or negative sentiment about the subject being discussed?

What can stop US Postal Service trucks? The inexorable march of time

The ageing fleet of delivery vehicles is long past due an overhaul. Among the common-sense upgrades employees want: air conditioning and more workspace

For the better part of the last 30 years, the flatulent buzz of the US Postal Service's boxy delivery vans - audible as they lighted from mailbox to mailbox - has been a familiar sound to most Americans. Neither snow nor rain nor heat nor gloom of night stays these trucks - but time, it turns out, will. Photograph: Bill Silas/AP

Neither snow nor rain nor heat nor gloom of night stays these trucks - but time, it turns out, will. Photograph: Bill Silas/AP
Prediction tasks involving text

What is the category/subject/topic of this article?

Apple Is Forming an Auto Team

By BRIAN X. CHEN and MIKE ISAAC  FEB. 19, 2015

SAN FRANCISCO — While Apple has been preparing to release its first wearable computers, the company has also been busy assembling a team to work on an automobile.

The company has collected about 200 people over the last few years — both from inside Apple and potential competitors like Tesla — to develop technologies for an electric car, according to two people with knowledge of the company’s plans, who asked not to be named because the plans were private.

The car project is still in its prototype phase, one person said, meaning it is probably many years away from being a viable product and might never reach the mass market if the quality of the vehicle fails to impress Apple’s executives.

It could also go nowhere if Apple struggles to find a compelling business opportunity in automobiles, a business that typically has much lower sales margins than

Which of these articles are relevant to my interests?
Prediction tasks involving text

Find me articles similar to this one

Meatloaf That Conquers the Mundane

I was raised on Midwestern meatloaf. My mother’s dependable recipe did not vary:
Ground beef, grated onion and carrot and a little
oatmeal were the main ingredients, along with a
dash of “seasoned salt.” A ribbon of bottled chili
sauce ran down a gully in the center.

Served hot, accompanied by Tater Tots, it was
dinner. Served cold for lunch, it was always a
sandwich on white bread, with potato chips on
the side. It was usually moist and tasty but never
remarkable, and there was no way you could call
it anything but meatloaf.

Do I harbor a kind of nostalgia for it? Yes. But
would I use that recipe now? I think not.

I have a friend from Brussels who loves to
entertain. Of his dinner party repertoire, one
dish is most requested and admired. It is pain de
veau, served with a vermouth-splashed mushroom sauce. In French, it
sounds elegant. Translated into English — veal loaf — it sounds dull.

The Italian word for meatloaf is polpettone. (Polpette are Italian meatballs;
polpettone are meatballs, too, but more diminutive.) This substantial
family-size meatball, whether slow or elongated, plain or fawzy, served
with tomato sauce or not, is beloved both in Italy and in Italian
communities throughout the world. Aside from its melodic, polysyllabic
name, polpettone is always well seasoned, prepared with case and served
with gravy.

It is usually a combination of different kinds
of ground meat, typically beef, pork and veal
meatballs, rolled cheese, and herbs.
Prediction tasks involving text

Which of these reviews am I most likely to agree with or find helpful?

Most Helpful Customer Reviews

1. Le Creuset on a budget
   By N. Lafond on October 24, 2007
   Enamelled cast iron cookware like this, was, until recently, only available from makers like Le Creuset. Lately, several lower cost makers have come on the scene, like Target and Iron-Craft. The new budget priced Lodge cookware is in the same price range as the low cost alternatives but completely outperforms them.
   I have all of the brands I have mentioned. The Lodge is the same weight as the Le Creuset which is much heavier than the other budget models. The edge where the lid and sides meet is a matte black porcelain on the Lodge and Le Creuset but is just exposed cast iron for the other budget models (which leads to rusting if you are not careful). The porcelain resists staining (even tomato sauces) in the Lodge and Le Creuset but the other budget models stain very easily. And finally, the Lodge and Le Creuset maintain a very polished interior finish that resists sticking which others do not. So, I see no performance differences at all between the Le Creuset and the Lodge whereas the comparably priced budget models are certainly inferior.
   If you plan of using these pots very heavily (every day for example) you might want to upgrade to the higher priced Lodge product. It has 4 coatings of enamel as opposed to 2 in this model. But if you use them once or twice a week I don't think you will need the added wear resistance.

2. OK pot, Great Price, Some Flaws.
   By J. G. Pavlovich on March 2, 2008
   This is a terrific value. The quality and performance match my Le Creuset pieces at a fraction of the price. The only slight design flaw I have found is that the rounded bottom makes browning large pieces of meat awkward. Other than that I have no complaints. Even heating. Easy clean up. I use it several times a week.
   UPDATE: I found a second minor problem. The inside rim of the lid has a couple of raised spots which prevent the lid from seating tightly. This causes steam to escape much faster than I would like during a long braise or stew.
Which of these sentences best summarizes people’s opinions?
‘Partridge in a Pear Tree’, brewed by ‘The Bruery’

Dark brown with a light tan head, minimal lace and low retention. Excellent aroma of dark fruit, plum, raisin and red grape with light vanilla, oak, caramel and toffee. Medium thick body with low carbonation. Flavor has strong brown sugar and molasses from the start over bready yeast and a dark fruit and plum finish. Minimal alcohol presence. Actually, this is a nice quad.
Using text to solve predictive tasks

• How to represent documents using features?
• Is text structured or unstructured?
• Does structure actually help us?
• How to account for the fact that most words may not convey much information?
• How can we find low-dimensional structure in text?
Web Mining and Recommender Systems

Bag-of-words models
We’d like a fixed-dimensional representation of documents, i.e., we’d like to describe them using **feature vectors**

This will allow us to compare documents, and associate weights with particular features to solve predictive tasks etc. (i.e., the kind of things we’ve been doing already)
Feature vectors from text

**Option 1:** just count how many times each word appears in each document

\[ F_{text} = [150, 0, 0, 0, 0, 0, 0, \ldots, 0] \]
Option 1: just count how many times each word appears in each document

Dark brown with a light tan head, minimal lace and low retention. Excellent aroma of dark fruit, plum, raisin and red grape with light vanilla, oak, caramel and toffee. Medium thick body with low carbonation. Flavor has strong brown sugar and molasses from the start over bready yeast and a dark fruit and plum finish. Minimal alcohol presence. Actually, this is a nice quad.

These two documents have exactly the same representation in this model, i.e., we’re completely ignoring syntax. This is called a “bag-of-words” model.
Option 1: just count how many times each word appears in each document

We’ve already seen some (potential) problems with this type of representation (dimensionality reduction), but let’s see what we can do to get it working
50,000 reviews are available on:
http://cseweb.ucsd.edu/classes/fa20/cse258-a/data/beer_50000.json
(see course webpage)

Code on course webpage
Feature vectors from text

**Q1:** How many words are there?

```python
wordCount = defaultdict(int)
for d in data:
    for w in d['review/text'].split():
        wordCount[w] += 1

print len(wordCount)
```

≈ 36k
Feature vectors from text

2: What if we remove capitalization/punctuation?

```python
wordCount = defaultdict(int)
punctuation = set(string.punctuation)
for d in data:
  for w in d['review/text'].split():
    w = ''.join([c for c in w.lower() if not c in punctuation])
    wordCount[w] += 1

print len(wordCount)
```

≈ 19k
3: What if we merge different inflections of words?

drinks → drink
drinking → drink
drinker → drink

argue → argu
arguing → argu
argues → argu
arguing → argu
argus → argu
3: What if we merge different inflections of words?

This process is called “stemming”

- The first stemmer was created by Julie Beth Lovins (in 1968!!)
- The most popular stemmer was created by Martin Porter in 1980
3: What if we merge different inflections of words? The algorithm is (fairly) simple but depends on a huge number of rules

What if we merge different inflections of words?

```python
wordCount = defaultdict(int)
punctuation = set(string.punctuation)
stemmer = nltk.stem.porter.PorterStemmer()
for d in data:
    for w in d['review/text'].split():
        w = ''.join([c for c in w.lower() if not c in punctuation])
        w = stemmer.stem(w)
        wordCount[w] += 1

print len(wordCount)
```

~15k
3: What if we merge different inflections of words?

• Stemming is **critical** for retrieval-type applications (e.g. we want Google to return pages with the word “cat” when we search for “cats”)
• Personally I tend not to use it for predictive tasks. Words like “waste” and “wasted” may have different meanings (in beer reviews), and we’re throwing that away by stemming
Feature vectors from text

4: Just discard extremely rare words...

```python
counts = [(wordCount[w], w) for w in wordCount]
counts.sort()
counts.reverse()

words = [x[1] for x in counts[:1000]]
```

- Pretty unsatisfying but at least we can get to some inference now!
Let’s do some inference!

**Problem 1: Sentiment analysis**

Let’s build a predictor of the form:

\[ f(\text{text}) \to \text{rating} \]

using a model based on linear regression:

\[ \text{rating} \approx \alpha + \sum_{w \in \text{text}} \text{count}(w) \cdot \theta_w \]

Code on course webpage
What do the parameters look like?

\[
\begin{align*}
\theta_{\text{fantastic}} &= 0.143 \\
\theta_{\text{watery}} &= -0.163 \\
\theta_{\text{and}} &= -0.008 \\
\theta_{\text{me}} &= -0.037
\end{align*}
\]
Why might parameters associated with “and”, “of”, etc. have non-zero values?

- Maybe they have meaning, in that they might frequently appear slightly more often in positive/negative phrases
- Or maybe we’re just measuring the length of the review…

How to fix this (and is it a problem)?
1) Add the length of the review to our feature vector
2) Remove stopwords
Feature vectors from text

Removing stopwords:

```python
from nltk.corpus import stopwords
stopwords.words("english")
```

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', 'her', 'hers', 'herself', 'it', 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', 'should', 'now']
Why remove stopwords?

some (potentially inconsistent) reasons:

• They convey little information, but are a substantial fraction of the corpus, so we can reduce our corpus size by ignoring them.
• They do convey information, but only by being correlated by a feature that we don’t want in our model.
• They make it more difficult to reason about which features are informative (e.g. they might make a model harder to visualize).
• We’re confounding their importance with that of phrases they appear in (e.g. words like “The Matrix”, “The Dark Night”, “The Hobbit” might predict that an article is about movies).

so use n-grams!
Feature vectors from text

We can build a richer predictor by using $n$-grams

e.g. “Medium thick body with low carbonation.”

unigrams: [“medium”, “thick”, “body”, “with”, “low”, “carbonation”]

bigrams: [“medium thick”, “thick body”, “body with”, “with low”, “low carbonation”]

trigrams: [“medium thick body”, “thick body with”, “body with low”, “with low carbonation”]

etc.
Feature vectors from text

We can build a richer predictor by using **n-grams**

- Fixes some of the issues associated with using a bag-of-words model – namely we recover some basic **syntax** – e.g. “good” and “not good” will have different weights associated with them in a sentiment model
- Increases the **dictionary size** by a lot, and increases the sparsity in the dictionary even further
- We might end up double (or triple-) counting some features (e.g. we’ll predict that “Adam Sandler”, “Adam”, and “Sandler” are associated with negative ratings, even though they’re all referring to the same concept)
We can build a richer predictor by using **n-grams**

- This last problem (that of double counting) is bigger than it seems: We’re **massively** increasing the number of features, but possibly increasing the number of **informative** features only slightly
- So, for a **fixed-length** representation (e.g. 1000 most-common words vs. 1000 most-common words+bigrams) the bigram model will quite possibly perform **worse** than the unigram model
Problem 2: Classification

Let’s build a predictor of the form:

\[ f(\text{text}) \rightarrow \text{class label} \]
So far...

Bags-of-words representations of text

- Stemming & stopwords
- Unigrams & N-grams
- Sentiment analysis & text classification

$$\text{rating} = O_o + \sum_{w \in D} \text{count}(w) \cdot O_w$$
Further reading:

• Original stemming paper
  “Development of a stemming algorithm” (Lovins, 1968):

• Porter’s paper on stemming
  “An algorithm for suffix stripping” (Porter, 1980):
Web Mining and Recommender Systems

TF-IDF
Learning Goals

- Introduce the concepts of Term Frequency and Document Frequency
- Discuss how to find "important" words in documents
- Build “item-to-item” recommenders based on text
When we studied recommender systems, we looked at:

- Approaches based on measuring similarity (cosine, jaccard, etc.)
- Approaches based on dimensionality reduction (latent factor models)

We’ll look at the same two concepts, but using textual representations
Finding relevant terms

So far we’ve dealt with huge vocabularies just by identifying the most frequently occurring words

But! The most informative words may be those that occur very rarely, e.g.:

- Proper nouns (e.g. people’s names) may predict the content of an article even though they show up rarely
- Extremely superlative (or extremely negative) language may appear rarely but be very predictive
Finding relevant terms

e.g. imagine applying something like cosine similarity to the document representations we’ve seen so far

e.g. are (the features of the reviews/IMDB descriptions of) these two documents “similar”, i.e., do they have high cosine similarity
Finding relevant terms

e.g. imagine applying something like cosine similarity to the document representations we’ve seen so far

\[ X_{bow} = [118, 50, \ldots] \]
\[ X_{pB} = [130, 75, \ldots] \]
Finding relevant terms

So how can we estimate the “relevance” of a word in a document?
e.g. which words in this document might help us to determine its content, or to find similar documents?

Despite Taylor making moves to end her long-standing feud with Katy, HollywoodLife.com has learned exclusively that Katy isn’t ready to let things go! Looks like the bad blood between Kat Perry, 29, and Taylor Swift, 25, is going to continue brewing. A source tells HollywoodLife.com exclusively that Katy prefers that their frenemy battle lines remain drawn, and we’ve got all the scoop on why Katy is set in her ways. Will these two ever bury the hatchet? Katy Perry & Taylor Swift Still Fighting?
“Taylor’s tried to reach out to make amends with Katy, but Katy is not going to accept it nor is she interested in having a friendship with Taylor,” a source tells HollywoodLife.com exclusively. “She wants nothing to do with Taylor. In Katy’s mind, Taylor shouldn’t even attempt to make a friendship happen. That ship has sailed.” While we love that Taylor has tried to end the feud, we can understand where Katy is coming from. If a friendship would ultimately never work, then why bother? These two have taken their feud everywhere from social media to magazines to the Super Bowl. Taylor’s managed to mend the fences with Katy’s BFF Diplo, but it looks like Taylor and Katy won’t be posing for pics together in the near future. Katy Perry & Taylor Swift: Their Drama Hits All-Time High At the very least, Katy and Taylor could tone down their feud. That’s not too much to ask, is it?
So how can we estimate the “relevance” of a word in a document? e.g. which words in this document might help us to determine its content, or to find similar documents?

Despite Taylor making moves to end her long-standing feud with Katy, HollywoodLife.com has learned exclusively that Katy isn’t ready to let things go! Looks like the bad blood between Kat Perry, 29, and Taylor Swift, 25, is going to continue brewing. A source tells HollywoodLife.com exclusively that Katy prefers that their frenemy battle lines remain drawn, and we’ve got all the scoop on why Katy isn’t ready to let things go! Will these two ever bury the hatchet? Katy Perry & Taylor Swift Still Fighting? “Taylor’s tried to reach out to make amends with Katy, but Katy is not going to accept it nor is she interested in a friendship with Taylor,” a source tells HollywoodLife.com exclusively. “She wants nothing to do with Taylor. In Katy’s mind, Taylor shouldn’t even attempt to make a friendship happen. That ship has sailed.” While we love that Taylor has tried to end the feud, we can understand where Katy is coming from. If a friendship would ultimately never work, then why bother? These two have taken their feud everywhere from social media to magazines to the Super Bowl. Taylor’s managed to mend the fences with Katy’s BFF Diplo, but it looks like Taylor and Katy won’t be posing for pics together in the near future. Katy Perry & Taylor Swift: Their Drama Hits All-Time High At the very least, Katy and Taylor could tone down their feud. That’s not too much to ask, right? We learned exclusively that the girls made desperate attempts to avoid each other at the very least, Katy and Taylor could tone down their feud. That’s not too much to ask, right?
Finding relevant terms

So how can we estimate the “relevance” of a word in a document?

e.g. which words in this document might help us to determine its content, or to find similar documents?

Despite Taylor making moves to end her long-standing feud with Katy, HollywoodLife.com has learned exclusively that Katy isn’t ready to let things go. Looks like the bad blood between Kat Perry, 29, and Taylor Swift, 25, is going to continue brewing. A source tells HollywoodLife.com exclusively that Katy prefers that their frenemy battle lines remain drawn, and we’ve got all the scoop on why Katy doesn’t want to put the hatchet down. Katy Perry & Taylor Swift Still Fighting?

“Taylor’s tried to reach out to make amends with Katy, but Katy is not going to accept it nor is she interested in having a friendship with Taylor,” a source tells HollywoodLife.com exclusively. “She wants nothing to do with Taylor. In Katy’s mind, Taylor shouldn’t even attempt to make a friendship happen. That ship has sailed.” While we love that Taylor has tried to end the feud, we can understand where Katy is coming from. If a friendship would ultimately never work, then why bother? These two have taken their feud everywhere from social media to magazines to the Super Bowl. Taylor’s managed to mend the fences with Katy’s BFF Diplo, but it looks like Taylor and Katy won’t be posing for pics together in the near future. Katy Perry & Taylor Swift: Their Drama Hits All-Time High At the very least, Katy and Taylor could tone down their feud. That’s not too much to ask, is it?
Finding relevant terms

So how can we estimate the “relevance” of a word in a document?

**Q:** The document discusses “the” more than it discusses “Taylor Swift”, so how might we come to the conclusion that “Taylor Swift” is the more relevant expression?

**A:** It discusses “the” **no more** than other documents do, but it discusses “Taylor Swift” **much more**
Finding relevant terms

Term frequency & document frequency

**Term frequency** ~ How much does the term appear in the document

**Inverse document frequency** ~ How “rare” is this term across all documents
Finding relevant terms

Term frequency & document frequency

\[ tf(t, d) = \# \text{times } t \text{ appears in } d \]

\[ df(t, D) = \# \text{docs in } D \text{ that contain } t \]
Finding relevant terms

Term frequency & document frequency

“Term frequency”: \( tf(t, d) = \) number of times the term \( t \) appears in the document \( d \)

\[
e.g. \; tf("Taylor Swift", \text{that news article}) = 3
\]

“Inverse document frequency“: \( idf(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|} \)  

\[
\text{term (e.g. "Taylor Swift")} \quad \text{set of documents}
\]

“Justification”: \( P(t \mid D) = \frac{|\{d \in D : t \in d\}|}{N} \) so \( idf(t, D) = -\log P(t \mid D) \)
Finding relevant terms

Term frequency & document frequency

**TF-IDF** is high → this word appears much more frequently in this document compared to other documents

**TF-IDF** is low → this word appears infrequently in this document, or it appears in many documents

\[
\text{tfidf}(t, d, D) = \text{tf}(t, d) \times \text{idf}(t, D)
\]
Finding relevant terms

Term frequency & document frequency

\( tf \) is sometimes defined differently, e.g.:

\[
tf'(t, d) = \delta(t \in d)
\]

\[
tf''(t, d) = \frac{\text{frequency of word}}{\text{frequency of most common word in document}}
\]

Both of these representations are invariant to the document length, compared to the regular definition which assigns higher weights to longer documents.
Finding relevant terms

How to use TF-IDF

• Frequently occurring words have little impact on the similarity
• The similarity is now determined by the words that are most “characteristic” of the document

[0,0,0.01,0,0.6,…,0.04,0,3,0,159.1,0]

[180.2,0,0.01,0.5,0,…,0.02,0,0.2,0,0,0]
Finding relevant terms

But what about when we’re **weighting** the parameters anyway?

\[ \text{MSE} + \lambda \| \Theta \|_2^2 \]

e.g. is:

\[
\text{rating} \approx \alpha + \sum_{w \in \text{text}} \text{count}(w) \cdot \theta_w
\]

really any different from:

\[
\text{rating} \approx \alpha + \sum_{w \in \text{text}} \text{tfidf}(w, d, D) \cdot \theta_w
\]

after we fit parameters?
Finding relevant terms

But what about when we’re weighting the parameters anyway?

Yes!

• The relative weights of features is different between documents, so the two representations are not the same (up to scale)

• When we regularize, the scale of the features matters – if some “unimportant” features are very large, then the model can overfit on them “for free”
Finding relevant terms

But what about when we’re weighting the parameters anyway?
Finding relevant terms

But what about when we’re **weighting** the parameters anyway?
Further reading:

- Original TF-IDF paper (from 1972)
  “A Statistical Interpretation of Term Specificity and Its Application in Retrieval”
  [http://goo.gl/1CLwUV](http://goo.gl/1CLwUV)
Web Mining and Recommender Systems

Dimensionality-reduction (latent factor) approaches to document representation
How can we find **low-dimensional structure** in documents?

**What we would like:**

- Action: action, loud, fast, explosion,
- Sci-fi: space, future, planet,

(review of “The Chronicles of Riddick”)

87 of 102 people found the following review helpful

★★★★★ You keep what you kill, December 27, 2004

By Sclinky “Sclinky” (Washington State) - See all my reviews

This review is from: The Chronicles of Riddick (Widescreen Unrated Director’s Cut) [DVD]

Even if I have to apologize to my friends and family, and my family, I have to admit that I really liked this movie. It’s a Sci-Fi movie with a ‘Mfax Maxx’ appeal that, while changing many things, left Riddick from ‘Pitch Black’ to be just Riddick. They did not change his attitude or soften him up or bring him out of his original character, which was very pleasing to ‘Pitch Black’ fans like myself.

First off, let me say that when playing the DVD, the first selection to come up is Conroy or Fight, and no explanation of the choices. This confused me at first, so I will mention off the bat that they are simply different menu formats, that each menu has the very same options, simply different background visuals. Select either one and continue with the movie.
Recall (from dimensionality reduction / recommender systems)

\[ R = \begin{pmatrix} 5 & 3 & \cdots & 1 \\ 4 & 2 & 1 \\ 3 & 1 & 3 \\ 2 & 2 & 4 \\ 1 & 5 & 2 \\ \vdots & \vdots & \vdots \\ 1 & 2 & \cdots & 1 \end{pmatrix} \]

\[ R = U\Sigma V^T \]

(square roots of) eigenvalues of \( RR^T \)

(eigenvectors of \( RR^T \))

(eigenvectors of \( R^T R \))

(matrix of ratings)
Taking the eigenvectors corresponding to the top-K eigenvalues is then the “best” rank-K approximation:

\[ R \approx U^{(k)} \Sigma^{(k)} V^{(k)T} \]

where

- \( R \) is the matrix to be approximated.
- \( U^{(k)} \) contains the top-K eigenvectors of \( R R^T \).
- \( \Sigma^{(k)} \) is a diagonal matrix containing the square roots of the top-K eigenvalues of \( R R^T \).
- \( V^{(k)T} \) contains the top-K eigenvectors of \( R^T R \).
Singular-value decomposition

What happens when we apply this to a matrix encoding our documents?

\[
X = \begin{pmatrix}
1 & 0 & \ldots & 4 \\
0 & 2 & 0 \\
31 & 23 & 97 \\
0 & 98 & 1 \\
473 & 88 & 347 \\
\vdots & \vdots & \vdots \\
11 & 34 & \ldots & 13
\end{pmatrix}
\]

\(X\) is a \(T \times D\) matrix whose columns are bag-of-words representations of our documents.

\(T = \) dictionary size
\(D = \) number of documents
Singular-value decomposition

What happens when we apply this to a matrix encoding our documents?

\[ X^T X \] is a \( D \times D \) matrix.

\[ U^{(k)} \sqrt{\sum (k)} \] is a low-rank approximation of each document eigenvectors of \( X^T X \)

\[ X X^T \] is a \( T \times T \) matrix.

\[ V^{(k)} \sqrt{\sum (k)} \] is a low-rank approximation of each term eigenvectors of \( X X^T \)
What happens when we apply this to a matrix encoding our documents?
Singular-value decomposition

What happens when we apply this to a matrix encoding our documents?

\[ X_{\text{Bow}} = \sum_{w} \sigma_{w} \cdot \mathbf{d} \]

- \( \mathbf{w} \) is the word representation
- \( \mathbf{D} \) is the document representation
- \( \sigma_{w} \) is the singular value
- \( \mathbf{d} \) is the representation of document \( d \)
- \( \mathbf{X}_{\text{Bow}} \) is the matrix encoding the documents

- \( \mathbf{X}_{\text{Bow}} \) represents the document word matrix

- \( \sigma_{w} \) represents the word singular value

- \( \mathbf{d} \) represents the document vector

- \( \mathbf{d} \) is the representation of the document
Singular-value decomposition

Using our low rank representation of each **document** we can...

- Compare two documents by their low dimensional representations (e.g. by cosine similarity)
- To retrieve a document (by first projecting the query into the low-dimensional document space)
- Cluster similar documents according to their low-dimensional representations
- Use the low-dimensional representation as features for some other prediction task

\[
\text{class} = \Theta_0 + \sum_k \Theta_{d,k} \cdot \Theta_k
\]
Singular-value decomposition

Using our low rank representation of each word we can...

• Identify potential synonyms – if two words have similar low-dimensional representations then they should have similar “roles” in documents and are potentially synonyms of each other

• This idea can even be applied across languages, where similar terms in different languages ought to have similar representations in parallel corpora of translated documents
This approach is called latent semantic analysis

- In practice, computing eigenvectors for matrices of the sizes in question is not practical – neither for $XX^T$ nor $X^TX$ (they won’t even fit in memory!)
- Instead one needs to resort to some approximation of the SVD, e.g. a method based on stochastic gradient descent that never requires us to compute $XX^T$ or $X^TX$ directly (much as we did when approximating rating matrices with low-rank terms)
Web Mining and Recommender Systems

word2vec
Goal: estimate the probability that a word appears near another (as opposed to Latent Semantic Analysis, which estimates a word count in a given document)

\[
\frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j}|w_t)
\]

- All tokens in document
- Context window of c adjacent words
- Probability that nearby word appears in the context of w_t
In practice, this probability is modeled approximately by trying to maximize the score of words that cooccur and minimizes the score of words that don't:

$$\log p(w_o | w_i) \approx \sigma(\gamma'_{w_o} \cdot \gamma_{w_i}) + \sum_{w \in \mathcal{N}} \log \sigma(-\gamma'_{w} \cdot \gamma_{w_i})$$

Co-occurring words should have compatible representations
Words that don't co-occur should have low compatibility

Note: Very similar to a latent factor model!
Word2vec

\[ \log p(w_o|w_i) \approx \sigma(\gamma'_w \cdot \gamma_{w_i}) + \sum_{w \in N} \log \sigma(-\gamma'_w \cdot \gamma_{w_i}) \]

Co-occurring words should have compatible representations

Words that don't co-occur should have low compatibility
Given its similarity to a latent factor representation, this idea has been adapted to use *item* sequences rather than *word* sequences.

\[ u_1 = [108, 15, 16, 21, \ldots ] \]
\[ u_2 = [15, 2, \ldots ] \]

\[ \mathbf{x}_i \cdot \mathbf{x}_j \text{ high if } i \text{ and } j \text{ appear nearby} \]
Item2vec vs FISM

FISM: \[ r(u,i) = \alpha + \beta_u + \beta_i + \sum_{j \in \text{neighbours}} x_{ij} \cdot \theta_i \]

Item2vec: \[ \sigma(\theta_i, \theta_j) \Rightarrow \text{high for nearby items} \]
\[ \Rightarrow \text{low for not nearby} \]
Given its similarity to a latent factor representation, this idea has been adapted to use *item* sequences rather than *word* sequences.

\[
\log p(i|j) \approx \sigma(\gamma'_i \cdot \gamma_j) + \sum_{i' \in \mathcal{N}} \log \sigma(-\gamma'_i \cdot \gamma_j)
\]

- **Probability that item \(i\) appears near \(j\)**
- **Repr. of item \(i\)**
- **Repr. of item \(j\)**
- **Random sample of negative items**
- **Items that don't co-occur should have low compatibility**
- **Co-occurring items should have compatible representations**
from gensim.models import Word2Vec

model = Word2Vec(reviewTokens, # Tokenized documents (list of lists)
                 min_count=5, # Minimum frequency before words are discarded
                 size=10, # Model dimensionality K
                 window=3, # Window size c
                 sg=1) # Skip-gram model (what I described)

model.wv.similar_by_word("grassy")

max \frac{\gamma_w \cdot \gamma_{\text{grassy}}}{\|\gamma_w\|\|\gamma_{\text{grassy}}\|} = \text{'citrus', 'citric', 'floral', 'flowery', 'piney', 'herbal'}
from gensim.models import Word2Vec

model = Word2Vec(itemSequences, # ordered sequences of items per user
                 min_count=5, # Minimum frequency before items are discarded
                 size=10, # Model dimensionality K
                 window=3, # Window size c
                 sg=1) # Skip-gram model (what I described)

model.wv.similar_by_word("Molson Canadian Light") # or really its itemID

Most similar items = 'Miller Light', 'Molson Golden',
'Piels', 'Coors Extra Gold', 'Labatt Canadian Ale' (etc.)
Word2Vec and Item2Vec in GenSim

- Note: this is a form of *item to item* recommendation, i.e., we learn which items appear in the context of other items, but there is no user representation.

- This is actually a very effective way to make recommendations based on a few items a user has consumed, without having to explicitly model the user.
Recommender Systems and NLP

Word2vec/item2vec is an example of a growing trend in recommender systems, where state-of-the-art models are often borrowed from NLP:

• NLP builds models of *documents* or *sentences*, which are just sequences of discrete tokens
• Typical tasks consist of predicting the next token, finding which words are semantically similar, etc.
• User interaction sequences are also just sequences of discrete tokens!
Word2vec/item2vec is an example of a growing trend in recommender systems, where state-of-the-art models are often borrowed from NLP:

- State of the art models for recommendation are (recently) based on tools like word2vec, LSTMs/RNNs, or self-attention/Transformer-like models
- See textbook for more details!
Web Mining and Recommender Systems

Case Study
Efficient Natural Language Response Suggestion for Smart Reply

MATTHEW HENDERSON, RAMI AL-RFOU, BRIAN STROPE, YUN-HSUAN SUNG, LÁSZLÓ LUKÁCS, RUIQI GUO, SANJIV KUMAR, BÁLINT MIKLÓS, and RAY KURZWEIL, Google

This paper presents a computationally efficient machine-learned method for natural language response suggestion. Feed-forward neural networks using n-gram embedding features encode messages into vectors which are optimized to give message-response pairs a high dot-product value. An optimized search finds response suggestions. The method is evaluated in a large-scale commercial e-mail application, Inbox by Gmail. Compared to a sequence-to-sequence approach, the new system achieves the same quality at a small fraction of the computational requirements and latency.

Additional Key Words and Phrases: Natural Language Understanding; Deep Learning; Semantics; Email

1 INTRODUCTION

Applications of natural language understanding (NLU) are becoming increasingly interesting with scalable machine learning, web-scale training datasets, and applications that enable fast and nuanced quality evaluations with large numbers of user interactions.

Early NLU systems parsed natural language with hand-crafted rules to explicit semantic representations, and used manually written state machines to generate specific responses from the output of parsing [18]. Such systems are generally limited to the situations imagined by the designer, and much of the development work involves writing more rules to improve the robustness of semantic understanding. The method presented here is trained on natural language data.
Efficient Natural Language Response Suggestion for Smart Reply

**Goal:** Automatically suggest common responses to e-mails
Efficient Natural Language Response Suggestion for Smart Reply

Basic setup

- If a new email $x$ is received, the system checks for trigger suggestions.
  - If no trigger suggestions are found, no smart reply suggestions are made.
  - If trigger suggestions are found, the response selection $(y_1, \ldots, y_k)$ is performed.
- Diversification $(y_{i_1}, \ldots, y_{i_m})$ is applied.
- The set $R$ is then selected and clustered.
- Smart reply suggestions are shown.
Efficient Natural Language Response Suggestion for Smart Reply

Previous solution (KDD 2016)

- Based on a seq2seq method

\[ P(y | x) = P(y_1, \ldots, y_n | x_1, \ldots, x_m) = \prod_{i=1}^{n} \text{LSTM}(y_i | x_1, \ldots, x_m, y_1, \ldots, y_{i-1}) \]
Efficient Natural Language Response Suggestion for Smart Reply

**Idea:** Replace this (complex) solution with a simple multiclass classification-based solution

\[ P(y \mid x) = \frac{P(x, y)}{\sum_k P(x, y_k)} \]

\[ P(x, y) \propto e^{S(x, y)} \]
Idea: Replace this (complex) solution with a simple multiclass classification-based solution

\[ P_{\text{approx}}(y \mid x) = \frac{e^{S(x,y)}}{\sum_{k=1}^{K} e^{S(x,y_k)}} \]
Model: $S(x,y)$

$\Psi(x) \in \mathbb{R}^d$
Efficient Natural Language Response Suggestion for Smart Reply

**Model:** Architecture v1

\[ S(x, y) = Wh \]

ReLU layer

ReLU layer

ReLU layer

\[ \Psi(x) \quad \Psi(y) \]
Efficient Natural Language Response Suggestion for Smart Reply

Model: Architecture v2
Efficient Natural Language Response Suggestion for Smart Reply

**Model:** Extensions

\[ S(x, y) = Wh \]

ReLU layer

ReLU layer

ReLU layer

ReLU layer

\[ \oplus_{i=1}^{M} h^i \]

\[ S(x', y) = W'h' \]

ReLU layer

ReLU layer

ReLU layer

\[ \Psi(x^i) \quad \Psi(y) \]
**Model:** Extensions

<table>
<thead>
<tr>
<th>Message: Did you manage to print the document?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>With response bias</strong></td>
</tr>
<tr>
<td>– Yes, I did.</td>
</tr>
<tr>
<td>– Yes, it’s done.</td>
</tr>
<tr>
<td>– No, I didn’t.</td>
</tr>
</tbody>
</table>
Efficient Natural Language Response Suggestion for Smart Reply

**Experiments:** (offline)

<table>
<thead>
<tr>
<th>Batch Size</th>
<th>Scoring Model</th>
<th>P@1</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>Joint</td>
<td>49%</td>
</tr>
<tr>
<td>25</td>
<td>Dot-product</td>
<td>48%</td>
</tr>
<tr>
<td>50</td>
<td>Dot-product</td>
<td>52%</td>
</tr>
</tbody>
</table>
### Experiments: (online)

<table>
<thead>
<tr>
<th>System</th>
<th>Experiment</th>
<th>Conversion rate relative to Seq2Seq</th>
<th>Latency relative to Seq2Seq</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exhaustive search</td>
<td>Use a joint scoring model to score all responses in $R$.</td>
<td>–</td>
<td>500%</td>
</tr>
<tr>
<td>Two pass</td>
<td>Two passes: dot-product then joint scoring.</td>
<td>67%</td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>Include response bias.</td>
<td>88%</td>
<td>10%</td>
</tr>
<tr>
<td></td>
<td>Improve sampling of dataset, and use multi-loss structure.</td>
<td>104%</td>
<td>10%</td>
</tr>
<tr>
<td>Single pass</td>
<td>Remove second pass.</td>
<td>104%</td>
<td>2%</td>
</tr>
<tr>
<td></td>
<td>Use hierarchical quantization for search.</td>
<td>104%</td>
<td>1%</td>
</tr>
</tbody>
</table>
Morals:

• Even a seemingly complex problem like natural-language response generation can be cast as a multiclass classification problem!
• Even a simple bag-of-words model proved to be sufficient, no need to handle “grammar” etc.
• Also, no personalization (though to what extent would this be possible with the data available?)
Building recommender systems based on the paradigm of conversation is a hot emerging topic. What is a "conversational paradigm"?

- Humans could interact via a **free-form dialog**, and occasionally recommend stuff to each other
- One human could recommend, another could **criticize/critique** the recommendations
- One person could provide **constraints**, which the other tries to satisfy
- Could simply be a form of recommendation that is more **interactive**
Query refinement (e.g. Mahmood and Ricci)

During several rounds, ask users "questions" that will help to get more information from them
- Questions could include asking about constraints, having users tell us what is "wrong" with recommendations, or telling us what they like
- Developing a "conversational strategy" then consists of choosing what type of questions to ask at each step
Query refinement (e.g. Mahmood and Ricci)

During several rounds, ask users what is wrong with their
Interactive recommendation (Christakopoulou et al.)

Not really "conversation" at all, more like interactive recommendation

- What "questions" can we ask to a (new) user to most quickly get information out of them?
- Questions are just evaluations of specific candidates, in a way that's designed to gather the most information
- E.g. should I ask about movies I think you like, movies that I think you'll know about, movies with the most uncertainty (etc.)?
- Challenge is to try and gather information while asking questions the user is capable of answering
Free-form conversation (Li et al.)

Closer to "real" conversation, though not totally free-form

• Uses mechanical turk to harvest conversations between two humans, who are given roles of "expert" and "seeker"
• Expert is given a set of movies, one of which is relevant to the seeker; the expert has to guess which one is relevant by asking questions
• Expert can either engage in a dialog turn or guess a movie
• During conversation turns, feedback can be given on items
• Following this, a reinforcement learning algorithm is trained to reproduce conversational steps; goal is to reach the goal item in as few turns as possible
Free-form conversation (Li et al.)

(see more in textbook)
Web Mining and Recommender Systems

Personalized Models of Text
Personalized Language Generation & Explanation

have:

\[ f(u, i) : U \times I \rightarrow \{1, 2, 3, 4, 5\} \]

want:

"Machine washable, all natural fabrics, true to size"
Generative models of text

Standard generative RNN, and encoder-decoder RNN

Can be used to generate reviews, including *personalized* generation

(see e.g. “Learning to generate reviews and discovering sentiment”, Radford et al. 2017)
Poured from 12oz bottle into half-liter Pilsner Urquell branded pilsner glass. **Appearance:** Pours a cloudy golden-orange color with a small, quickly dissipating white head that leaves a bit of lace behind. **Smell:** Smells HEAVILY of citrus. By heavily, I mean that this smells like kitchen cleaner with added wheat. **Taste:** Tastes heavily of citrus—lemon, lime, and orange with a hint of wheat at the end. Mouthfeel: Thin, with a bit too much carbonation. Refreshing. **Drinkability:** If I wanted lemonade, then I would have bought that.

Poured from a 12oz bottle into a 16oz Samuel Adams Perfect Pint glass. **Appearance:** Very pale golden color with a thin, white head that leaves little lacing. **Smell:** Very mild and inoffensive aromas of citrus. **Taste:** Starts with the same tastes of the citrus and fruit flavors of orange and lemon and the orange taste is all there. There is a little bit of wheat that is pretty weak, but it is sort of harsh (in a good way) and ends with a slightly bitter aftertaste. **Mouthfeel:** Light body with a little alcohol burn. Finish is slightly dry with some lingering spice. **Drinkability:** A decent beer, but not great. I don’t think I would rate this anytime soon as it says that there are other Belgian beers out there, but this is a good choice for a warm day when it’s always available in the North Coast Brewing Company party.
Use-cases for personalized generative models of text

Why would we want to generate personalized text?

- Directly generating reviews may not be the most compelling application!
- But having a high-fidelity personalize language model could be used to:
  a) Build other tools that require personalized language (e.g. dialog systems, assistive writing tools)
  b) Explain machine predictions to users (in a personalized way)
  c) Develop new modes of recommendation (e.g. conversational interfaces)
Use-cases for personalized generative models of text

More in textbook:

• Using text to improve the **accuracy** of recommender systems
• Using text to make recommender systems more **explainable**
• Recommender systems based on the paradigm of **natural language conversation**
• Recommender systems based on more complex NLP models (e.g. transformer/LSTM)